2.28 DETAILED RESULTS FOR RANGE TRACK

No significant errors were found in the Range Track Functional Element for ESAMS 2.6.2. The overall code quality is good, with only a few corrections recommended for the internal documentation. External documentation for ESAMS 2.5 is adequate-for this FE as implemented in ESAMS 2.6.2.

The table listed below summarizes the desk-checking and software testing verification activities for each subroutine in the Range Track Functional Element. The two results columns contain checks if no discrepancies were found.

TABLE 2.28-1. Verification Results Summary.

Design Element	Code Location	Desk Check Result	Test Case ID	Test Case Result
28-1 Pulse Edge Positions	PRTION	√	28-5,6	√
28-2 Early/Late Gate Signal Amplitude	INTGRT	√	28-7	√
28-3 Range Gate Error	RNGDSC	√	28-1,2, 3,4	√
28-4 Range Gate Servo Dynamics	SVORNG	√	28-4	√

2.28.1 Overview

In pulsed systems, range tracking is the process of continuously measuring the time delay between transmission of an RF pulse and the reception of the pulse echo from the target. The range measurement is the most precise position-coordinate measurement of the radar system. In addition, range tracking provides an important means of multiple target discrimination by eliminating from the receiver train signal returns other than those of the intended target. This is accomplished by positioning the range gate to correspond to a time period when the pulse return is expected and closed the remainder of the time. The range of the target is tracked in a closed-loop fashion by generating range gate position error measurements. These range errors are produced by dividing the gate into early and late segments, then measuring and comparing the amount of pulse return energy that falls within each gate. This error term is then used to reposition the range gate so that it is centered on the target.

ESAMS 2.6.2 implementation of Range Track is accomplished with four primary subroutines and two support subroutines. Subroutines PRTION, INTGRT, RNGDSC and SVORNG are used exclusively for range tracking operations. Subroutines HEAPIN and HEAPSR are support routines which sort arrays. The six subroutines used for this FE are described in Table 2.28-2.

TABLE 2.28-2. Range Track Subroutine Descriptions.

Module Name	Description
RNGDSC	Computes the range track error as determined by a split-gate tracking algorithm. The early and late gates are weighted by the respective weighing factors, and the energy content difference between the early and late gates is used to compute a range error signal.
PRTION	Partitions the range gate into a series of intervals defined by the edge points of the pulses within the gate the edges of the range gate itself.
INTGRT	Calculates the signal level in the early and late gate portions of the range gate one interval at a time.
SVORNG	Computes the range gate servo response from current and previous input and output commands.
HEAPSR	Sorts an array into a heap.
HEAPIN	Adds a new element to any array and rearranges it into a heap.

2.28.2 Verification Design Elements

Design elements defined for the Range Track FE are listed in Table 2.28-3; they are fully described in Section 2.28.2 of ASP II. A design element is an algorithm that represents a specific component of the FE design.

TABLE 2.28-3. Range Track Design Elements.

Subroutine	Design Element	Description
PRTION	28-1 Pulse Edge Positions	Calculation of pulse edge positions in relation to left and right edges of the gate.
INTGRT	28-2 Early/Late Gate Signal Amplitude	Sums the signals by interval to determine the signal strength within the early and late portions of the range gate.
RNGDSC	28-3 Range Gate Error	Calculation of range error using difference-to-sum ratio of early and late gate signals.
SVORNG	28-4 Range Gate Servo Dynamics	Computes range gate servo response to range error derived input command.

2.28.3 Desk Checking Activities and Results

The code implementing this FE was manually examined using the procedures described in Section 1.1 of this report. No code discrepancies were discovered.

Except as noted in Table 2.28-4 below, overall code quality and internal documentation were evaluated as good. Subroutine I/O and logical flow were found to match the ASP II descriptions.

TABLE 2.28-4. Code Quality and Internal Documentation Results.

Subroutine	Code Quality	Internal Documentation
SVORNG	OK	1. The variable DELTAT is in the argument list but it is not used. 2. The subroutine contains no prologue

2.28.4 Software Test Cases and Results

All subroutines implementing the range tracking functional element were tested by off-line and integrated code. Off-line testing was performed using copies of ESAMS code run on a PC. For integrated testing, the entire ESAMS model was run in debug mode. Unless otherwise indicated, the standard ESAMS data files for the systems under consideration were used as input for all test cases.

TABLE 2.28-5. Range Tracking Software Test Cases.

Test Case ID	Test Case Description
28-1	Objective: Check range gate error computation in both directions. Procedure: 1. Write an off-line driver to run subroutine RNGDSC with the following data set. 2. Stop in subroutine RNGDSC 3. Stop on line 128 4. Examine variable RERROR 5. Compare to pre-calculated values Target range = 11000.1 Target pulse width = 2.9E-05 Range gate center = 11000.0 Range gate width = 7.0E-05 Verify: Alarm values match independently calculated values. Result: OK
28-2	Objective: Check track-lost logic when target does not fall within the range gate. Procedure: 1. Write an off-line driver to run subroutine RNGDSC with the following data set. 2. Stop in subroutine RNGDSC 3. Stop on line 81 4. Step to next executable line of code (should be line 119) to check that branching executes correctly 5. Stop on line 128 6. Examine variables RERROR, RGATE, RTSI and CSTTIM for reasonableness 7. Go to 4 (loop until CSTTIM exceeds maximum coast time) Target range = 23000.0 Target pulse width = 2.9E-05 Target velocity = -1000.0 Range gate center = 22000.0 Range gate width = 7.0E-05 Verify: 1. Next line executed after line 81 is 119. 2. ALARM values are reasonable. Result: OK

TABLE 2.28-5. Range Tracking Software Test Cases. (Contd.)

Test Case ID	Test Case Description
28-3	Objective: Check track-lost logic when target is initially outside of the gate. Procedure: 1. Write an off-line driver to run subroutine RNGDSC with the following data set. 2. Stop in subroutine RNGDSC. 3. Stop on line 128. 4. Examine variables RERROR, RGATE and RTSI for reasonableness. 5. Go to 4. (loop over 50 iterations) Target range = 21000.0 Target pulse width = 2.9E-05 Target velocity = 1000.0 Range gate center = 22000.0 Range gate width = 7.0E-05 Verify: 1. ALARM values are reasonable. Result: OK
28-4	Objective: Check range gate motion in both directions. Procedure: 1. Write an off-line driver to run subroutine RNGDSC with the following data sets. 2. Stop in subroutine RNGDSC. 3. Stop on line 128. 4. Examine variables RERROR, RGATE and RTSI for reasonableness. 5. Go to 2. (loop over 10 iterations) 6. Repeat for each of the three cases. Case 1: Target range = 21000.0 Target pulse width = 2.9E-05 Target velocity = 1000.0 Range gate center = 22000.0 Range gate width = 7.0E-05 Case 2: Target range = 21000.0 Target pulse width = 2.9E-05 Target velocity = 1000.0 Range gate center = 22000.0 Range gate width = 7.0E-05 Case 3: Target range = 21000.0 Target pulse width = 2.9E-05 Target velocity = 1000.0 Range gate center = 22000.0 Range gate width = 7.0E-05 Verify: 1. ALARM values are reasonable. Result: OK

TABLE 2.28-5. Range Tracking Software Test Cases. (Contd.)

Test Case ID	Test Case Description
28-5	Objective: Check range gate boundary calculations. Procedure: 1. Set target and range gate data: 2. Stop in subroutine PRTION. 3. Stop on line 78. 4. Examine variables TRGC, TRGL and TRGR. 5. Compare with pre-calculated values. Target range = 23003.0 Target pulse width = 2.9E-05 Pulse magnitude = 1.0 Range gate center = 23000.0 Range gate width = 7.0E-05 Verify: ALARM values match independently calculated values. Result: OK
28-6	Objective: Check signal pulse end-point computations for three overlapping target signals. Procedure: 1. Set target and range gate data: 2. Stop in subroutine PRTION. 3. Stop on line 111. 4. Examine variable array PTAR. 5. Compare with pre-calculated values (check for correct array location). 6. Stop on line 165. 7. Examine variable array TRGEP. 8. Compare with pre-calculated values (check for correct order). Signal 1: Target range = 23003.0 Target pulse width = 2.9E-05 Pulse magnitude = 1.0 Signal 2: Target range = 23001.0 Target pulse width = 2.0E-05 Pulse magnitude = 1.0 Signal 3: Target range = 23002.0 Target pulse width = 2.0E-05 Pulse magnitude = 1.0 Range gate center = 23000.0 Range gate width = 7.0E-05 Verify: ALARM values match independently calculated values. Result: OK

TABLE 2.28-5. Range Tracking Software Test Cases. (Contd.)

Test Case ID	Test Case Description
28-7	Objective: Check signal envelope and integration calculations. Procedure: 1. Set target and range gate data: 2. Stop in subroutine PRTION. 3. Stop on line 109. 4. Examine variable arrays ASGM2 and DLEP. 5. Compare with pre-calculated values. 6. Stop on line 127. 7. Examine variable IEPC. 8. Compare with pre-calculated index value. 9. Stop on line 173. 10. Examine EARLY and LATE. 11. Compare with pre-calculated values. Signal 1: Target range = 23003.0 Target pulse width = 2.9E-05 Pulse magnitude = 1.0 Signal 2: Target range = 23001.0 Target pulse width = 2.0E-05 Pulse magnitude = 1.0 Signal 3: Target range = 23002.0 Target pulse width = 2.0E-05 Pulse magnitude = 1.0 Range gate center = 23000.0 Range gate width = 7.0E-05 Verify: ALARM values match independently calculated values. Result: OK

2.28.5 Conclusions and Recommendations

2.28.5.1 Code Discrepancies

There were no code discrepancies uncovered in Range Track FE for ESAMS 2.6.2.

2.28.5.2 Code Quality and Internal Documentation

The quality of the code for the Range Track FE in ESAMS 2.6.2 is generally good. Internal documentation is generally good except for subroutine SVORNG which has an error in the argument list as well as no prologue, i.e., abstract, purpose, etc.

2.28.5.3 External Documentation

There is no external documentation for ESAMS 2.6.2. Therefore, the external documentation for ESAMS 2.5 was used. Other than choosing which missile/radar to use, there is no direct user interface to range track FE, therefore, it is not discussed in the *User's Manual* [3]. The *Analyst's Manual* [4] contains an adequate, although upper level explanation of range tracking methodology.